# How To Create A Rest Api With SAP Abap And Apply MVC1 Routing

**beyhan_meyrali**
Active Contributor

2022 Nov 18 7:33 AM

15 Kudos

48,052 Views

Hi,

In this blog post, I would like show how to create Rest api and how to apply MVC1 routing to handle different request simply from a controller class.

For that, first we will create handler and controller class for rest structure. Then we will add mvc1 controller class and model class to process business logic.

And finally we will create a service to handle rest requests.

At the end of the post, there are web browser, postman and abap consuming examples for the same rest api.

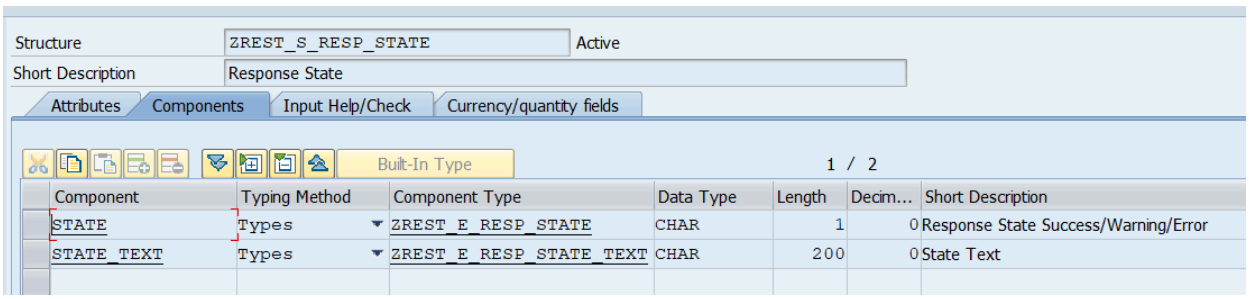
To read more about SAP REST, have a look at REST Tutorial.


Let's start.

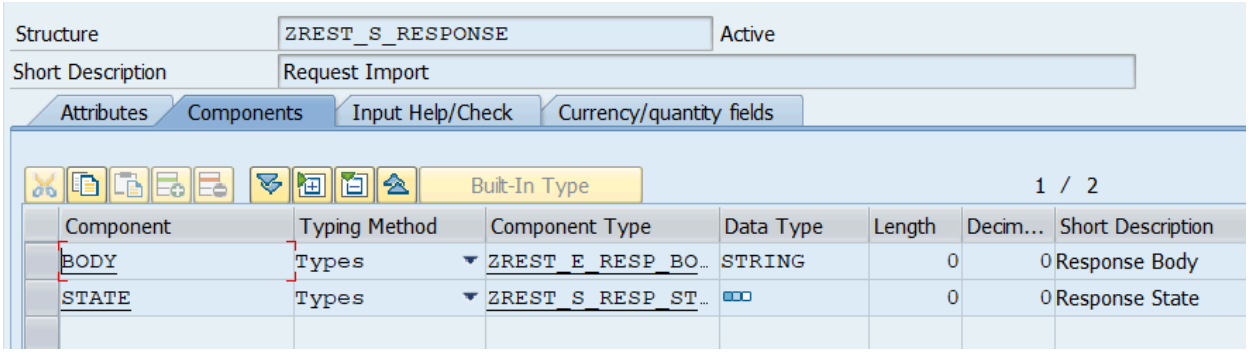
**Create following structures**;

- ZREST_S_RESP_STATE
- ZREST_S_RESPONSE
- ZREST_S_REQUEST

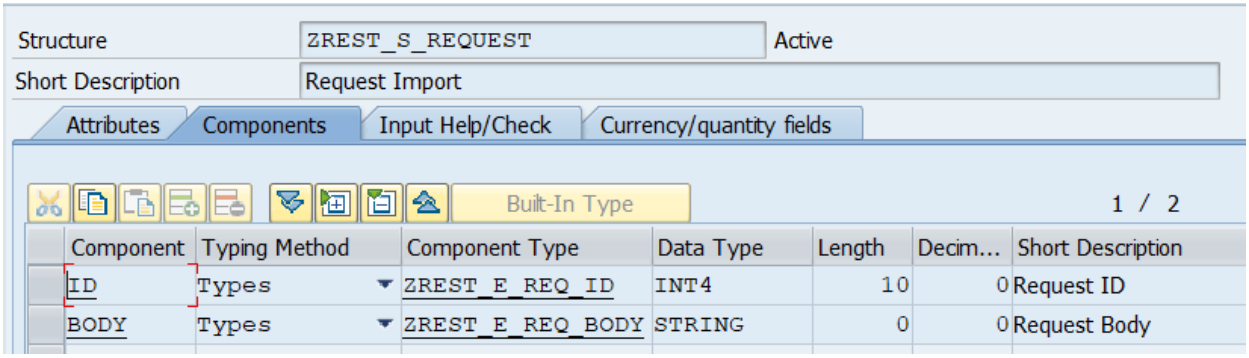
- ZREST_S_RESP_STATE



- ZREST_S_RESPONSE




- ZREST_S_REQUEST




Now we will **create classes.**

Call Stack for a call

ABAP and Screen Stack

| Sta... | Stac... | S.. | Event Type | Event | Program |
|---|---|---|---|---|---|
| ⇨ | 13 | | METHOD | GET_DATETIME | ZREST_CL_MODEL==========CP |
| | 12 | | METHOD | PROCESS_REQUEST | ZREST_CL_REQ_CONTROLLER=======CP |
| | 11 | | METHOD | IF_REST_RESOURCE~POST | ZREST_CL_REQ_HTTP_HANDLER=====CP |
| | 10 | | METHOD | DO_HANDLE | CL_REST_RESOURCE==========CP |
| | 9 | | METHOD | DO_HANDLE_CONDITIONAL | CL_REST_RESOURCE==========CP |
| | 8 | | METHOD | IF_REST_HANDLER~HANDLE | CL_REST_RESOURCE==========CP |
| | 7 | | METHOD | IF_REST_HANDLER~HANDLE | CL_REST_ROUTER==========CP |
| | 6 | | METHOD | IF_HTTP_EXTENSION~HANDLE_REQUE... | CL_REST_HTTP_HANDLER==========CP |
| | 5 | | METHOD | EXECUTE_REQUEST | CL_HTTP_SERVER==========CP |
| | 4 | | FUNCTION | HTTP_DISPATCH_REQUEST | SAPLHTTP_RUNTIME |
| | 3 | | MODULE (PBO) | %_HTTP_START | SAPMHTTP |
| | 2 | | PBO SCREEN | 0010 | SAPMHTTP |
| | 1 | | TRANSACTION | () | |

## Classes

- ZREST_CL_DEFS
- ZREST_CL_MODEL
- ZREST_CL_REQ_CONTROLLER
- ZREST_CL_REQ_HTTP_HANDLER
- ZREST_CL_HTTP_HANDLER

```
CLASS zrest_cl_defs DEFINITION
  PUBLIC
  CREATE PUBLIC .

  PUBLIC SECTION.

    CONSTANTS c_state_success TYPE char1 VALUE 'S' ##NO_TEXT.
    CONSTANTS c_state_warning TYPE char1 VALUE 'W' ##NO_TEXT.
    CONSTANTS c_state_error TYPE char1 VALUE 'E' ##NO_TEXT.

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS ZREST_CL_DEFS IMPLEMENTATION.
ENDCLASS.
```

```
CLASS zrest_cl_model DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    METHODS get_datetime
      EXPORTING
        !response_body TYPE zrest_s_response-body
        !state         TYPE zrest_s_response-state .

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS ZREST_CL_MODEL IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------
* | Instance Public Method ZREST_CL_MODEL->GET_DATETIME
* +--------------------------------------------------------------------
* | [<---] RESPONSE_BODY                  TYPE        ZREST_S_RESPONS
* | [<---] STATE                          TYPE        ZREST_S_RESPONS
* +--------------------------------------------------------------------
  METHOD get_datetime.
    DATA: exref TYPE REF TO cx_root.
    TRY .

        TYPES : BEGIN OF ty_res,
                  datetime TYPE tzntimestp,
                END OF ty_res.

        DATA: res TYPE ty_res.
        res-datetime = sy-datum && sy-uzeit.

        response_body = /ui2/cl_json=>serialize( EXPORTING data = res

        state-state = zrest_cl_defs=>c_state_success.
```

```
      CATCH cx_root INTO exref.
        state-state = zrest_cl_defs=>c_state_error.
        state-state_text = exref->get_text( ).
    ENDTRY.
  ENDMETHOD.
ENDCLASS.
```

```
CLASS zrest_cl_req_controller DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    METHODS process_request
      IMPORTING
        !req_json      TYPE string
      EXPORTING
        !response_json TYPE string
        !response_stc  TYPE zrest_s_response .

  PROTECTED SECTION.
  PRIVATE SECTION.

    CONSTANTS:
    c_req_get_datetime TYPE zrest_e_req_id VALUE '1001'.

    METHODS conv_stc_to_json
      IMPORTING
                response_stc  TYPE zrest_s_response
      RETURNING VALUE(result) TYPE string.

ENDCLASS.



CLASS ZREST_CL_REQ_CONTROLLER IMPLEMENTATION.


* <SIGNATURE>-------------------------------------------------------
* | Instance Private Method ZREST_CL_REQ_CONTROLLER->CONV_STC_TO_JSON
* +-------------------------------------------------------------------
* | [--->] RESPONSE_STC                   TYPE        ZREST_S_RESPONS
* | [<-()] RESULT                         TYPE        STRING
* +-------------------------------------------------------------------
  METHOD conv_stc_to_json.
    DATA: exref TYPE REF TO cx_root.
    TRY .
        result = /ui2/cl_json=>serialize( EXPORTING data = response_s
```

```
      CATCH cx_root.
        result = exref->get_text( ).
    ENDTRY.
  ENDMETHOD.


* <SIGNATURE>------------------------------------------------------------
* | Instance Public Method ZREST_CL_REQ_CONTROLLER->PROCESS_REQUEST
* +-----------------------------------------------------------------------
* | [--->] REQ_JSON                       TYPE        STRING
* | [<---] RESPONSE_JSON                  TYPE        STRING
* | [<---] RESPONSE_STC                   TYPE        ZREST_S_RESPONS
* +-----------------------------------------------------------------------
  METHOD process_request.
    DATA: exref TYPE REF TO cx_root.
    TRY .
        DATA req_stc TYPE zrest_s_request.
        /ui2/cl_json=>deserialize(
          EXPORTING
            json             = req_json
          CHANGING
            data             = req_stc
        ).

        IF req_stc-id IS NOT INITIAL.

          DATA(model) = NEW zrest_cl_model( ).

          IF req_stc-id EQ c_req_get_datetime.

            model->get_datetime(
              IMPORTING
                response_body = response_stc-body
                state         = response_stc-state
            ).

          ELSE.
            response_stc-state-state = zrest_cl_defs=>c_state_warning
            MESSAGE s001(zrest_msg) INTO response_stc-state-state_tex
          ENDIF.

        ELSE.
```

```
            "Fill dummy content as sample
            req_stc-id = 999999.
            req_stc-body = 'Some Json Content'.
            response_stc-body = /ui2/cl_json=>serialize( EXPORTING data

            response_stc-state-state = zrest_cl_defs=>c_state_warning.
            MESSAGE s002(zrest_msg) INTO response_stc-state-state_text.
          ENDIF.

          response_json = conv_stc_to_json( response_stc = response_stc

        CATCH cx_root.
          response_stc-state-state = zrest_cl_defs=>c_state_error.
          response_stc-state-state_text = exref->get_text( ).

          response_json = conv_stc_to_json( response_stc = response_stc
      ENDTRY.
    ENDMETHOD.
ENDCLASS.
```

```
CLASS zrest_cl_req_http_handler DEFINITION
  PUBLIC
  INHERITING FROM cl_rest_resource
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    METHODS if_rest_resource~get
        REDEFINITION .

    METHODS if_rest_resource~post
        REDEFINITION .

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS ZREST_CL_REQ_HTTP_HANDLER IMPLEMENTATION.


* <SIGNATURE>-------------------------------------------------------
* | Instance Public Method ZREST_CL_REQ_HTTP_HANDLER->IF_REST_RESOURC
* +-------------------------------------------------------------------
* +-------------------------------------------------------------------
  METHOD if_rest_resource~get.

    DATA(req_json) = mo_request->get_uri_query_parameter( iv_name = '

    DATA(controller) = NEW zrest_cl_req_controller( ).
    controller->process_request(
      EXPORTING
        req_json      = req_json
      IMPORTING
        response_json = DATA(response_json)
    ).

    mo_response->create_entity( )->set_string_data( iv_data = respons
  ENDMETHOD.
```

```
* <SIGNATURE>-------------------------------------------------------
* | Instance Public Method ZREST_CL_REQ_HTTP_HANDLER->IF_REST_RESOURC
* +-------------------------------------------------------------------
* | [--->] IO_ENTITY                      TYPE REF TO IF_REST_ENTITY
* +-------------------------------------------------------------------
  METHOD if_rest_resource~post.

    DATA(req_json) = mo_request->get_entity( )->get_string_data( ).

    DATA(controller) = NEW zrest_cl_req_controller( ).
    controller->process_request(
      EXPORTING
        req_json      = req_json
      IMPORTING
        response_json = DATA(response_json)
    ).

    mo_response->create_entity( )->set_string_data( iv_data = respons

  ENDMETHOD.
ENDCLASS.
```

CSRF is disabled in handler below. Disabling it from GUI Parameters of Service does not work. You need to implement HANDLE_CSRF_TOKEN in order to disable it for rest.

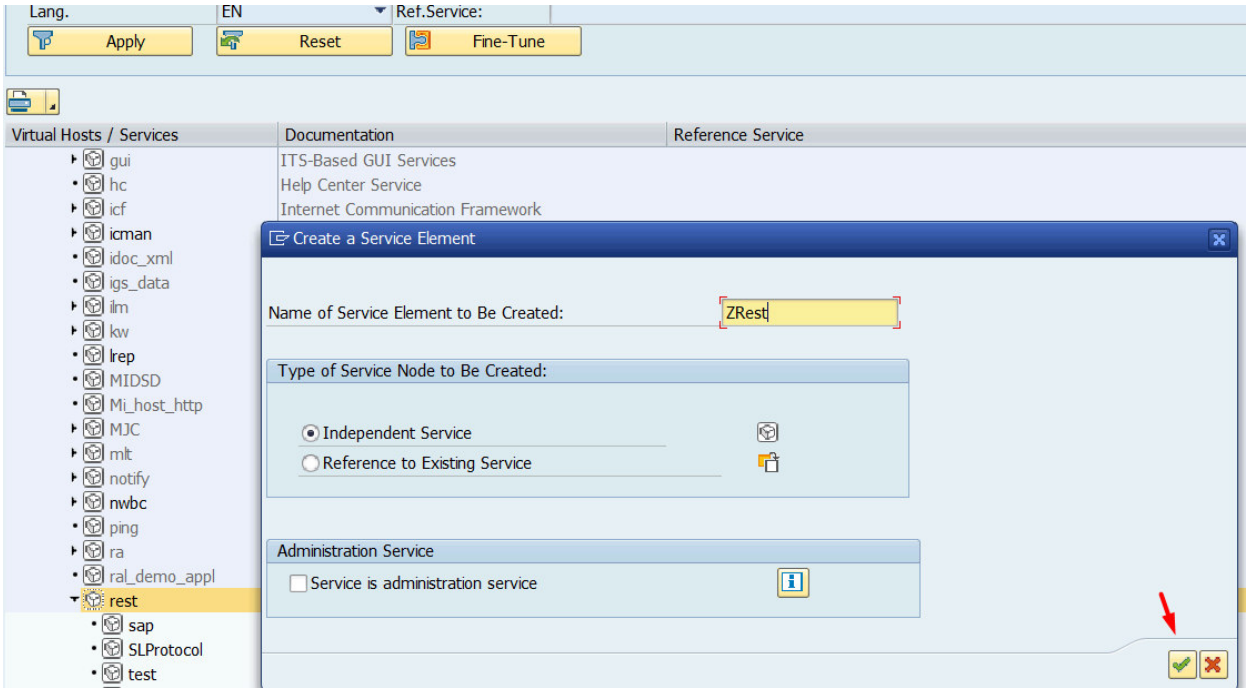
```
class ZREST_CL_HTTP_HANDLER definition
  public
  inheriting from CL_REST_HTTP_HANDLER
  create public .

public section.

  "Provides routing. Routing paths are assigned to controllers in thi
  methods IF_REST_APPLICATION~GET_ROOT_HANDLER
    redefinition .

protected section.

  "If you want to disable, redefine that method. Just as an empty met
  methods HANDLE_CSRF_TOKEN
    redefinition .

  PRIVATE SECTION.
ENDCLASS.



CLASS ZREST_CL_HTTP_HANDLER IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------
* | Instance Protected Method ZREST_CL_HTTP_HANDLER->HANDLE_CSRF_TOKE
* +--------------------------------------------------------------------
* | [--->] IO_CSRF_HANDLER                TYPE REF TO IF_REST_CSRF_HA
* | [--->] IO_REQUEST                     TYPE REF TO IF_REST_REQUEST
* | [--->] IO_RESPONSE                    TYPE REF TO IF_REST_RESPONS
* +--------------------------------------------------------------------
  method HANDLE_CSRF_TOKEN.
*CALL METHOD SUPER->HANDLE_CSRF_TOKEN
*  EXPORTING
*    IO_CSRF_HANDLER =
*    IO_REQUEST      =
*    IO_RESPONSE     =
*    .
  endmethod.
```

```
* <SIGNATURE>-------------------------------------------------------
* | Instance Public Method ZREST_CL_HTTP_HANDLER->IF_REST_APPLICATION
* +-------------------------------------------------------------------
* | [<-()] RO_ROOT_HANDLER                TYPE REF TO IF_REST_HANDLER
* +-------------------------------------------------------------------
  METHOD if_rest_application~get_root_handler.

    "Provides routing.
    "Service path /sap/bc/rest
    "Sample URL http://vhcalnplci:8000/sap/bc/rest/zrest/Rest?sap-cli
    DATA(root_handler) = NEW cl_rest_router( ).
    root_handler->attach(
      EXPORTING
        iv_template      = '/Rest'                " Unified Name for
        iv_handler_class = 'ZREST_CL_REQ_HTTP_HANDLER'
    ).

*    "You can add more request handler classes
*    "Service path /sap/bc/rest
*    "Sample URL http://vhcalnplci:8000/sap/bc/rest/zrest/Rest2?sap-c
*    root_handler->attach(
*      EXPORTING
*        iv_template      = '/Rest2'                " Unified Name f
*        iv_handler_class = 'ZREST_CL_REQ_HTTP_HANDLER2'
*    ).

    ro_root_handler = root_handler.
  ENDMETHOD.
ENDCLASS.
```

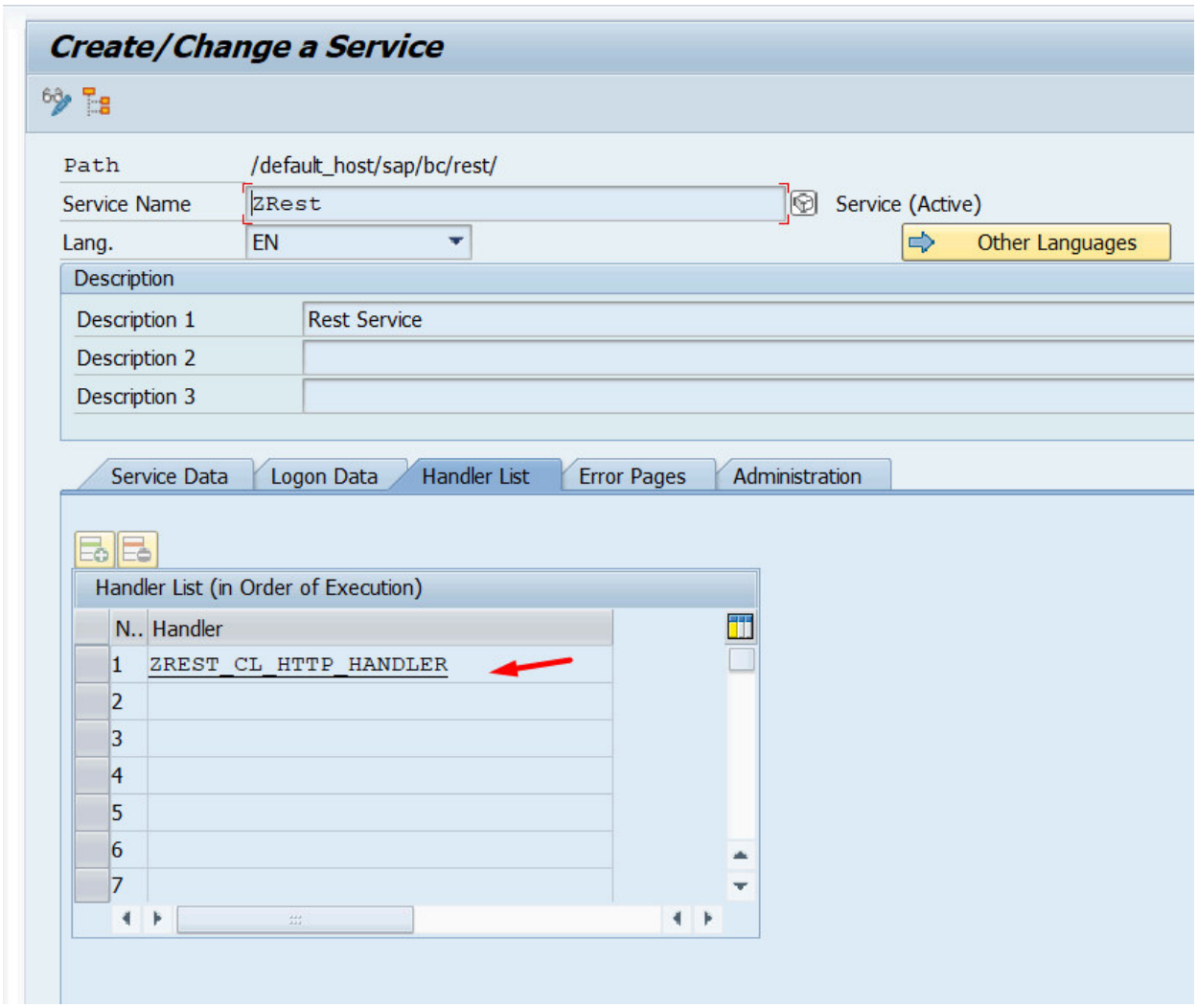
And final step, **create a service**.

Open SICF tcode and run.

Go to /sap/bc/rest and add new sub element




Add description and go to Handler List tab and our class, ZREST_CL_HTTP_HANDLER, as handler.



Activate Service. Right click service and click test. It will open browser. Change url to handle /Rest requests.

In my case, http://vhcalnplci:8000/sap/bc/rest/zrest/**Rest**

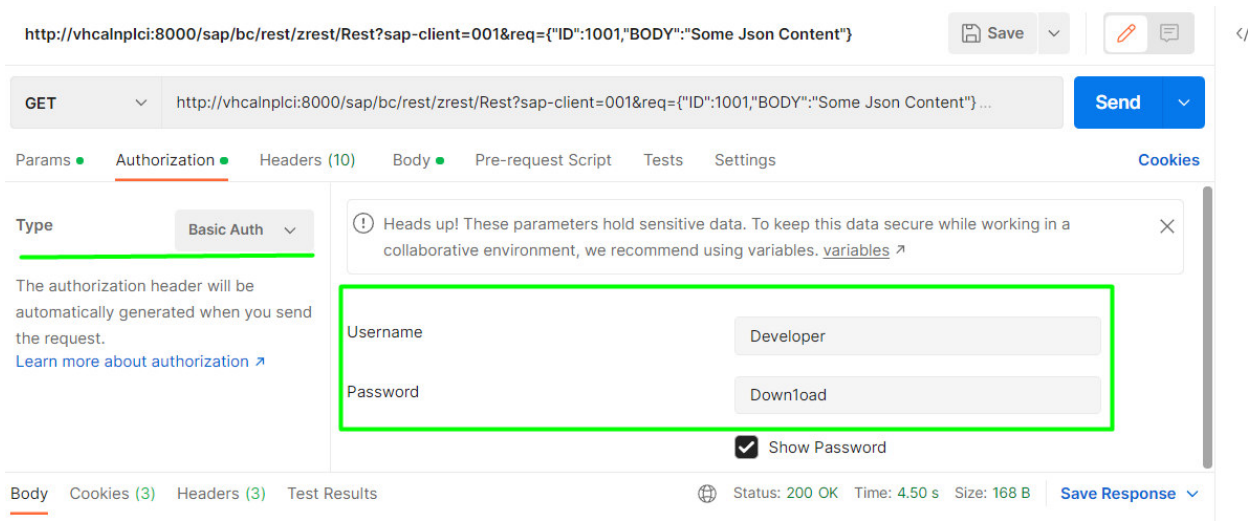
If you want to pass some params in get request, add query string parameters like 'http://vhcalnplci:8000/sap/bc/rest/zrest/Rest?sap-client=001&req={"ID":1001,"BODY":"Some Json Content"}'

If you do not disable CSRF on handler, you will have issues calling non-get methods, such as POST methods from POSTMAN or from a client different than server itself.

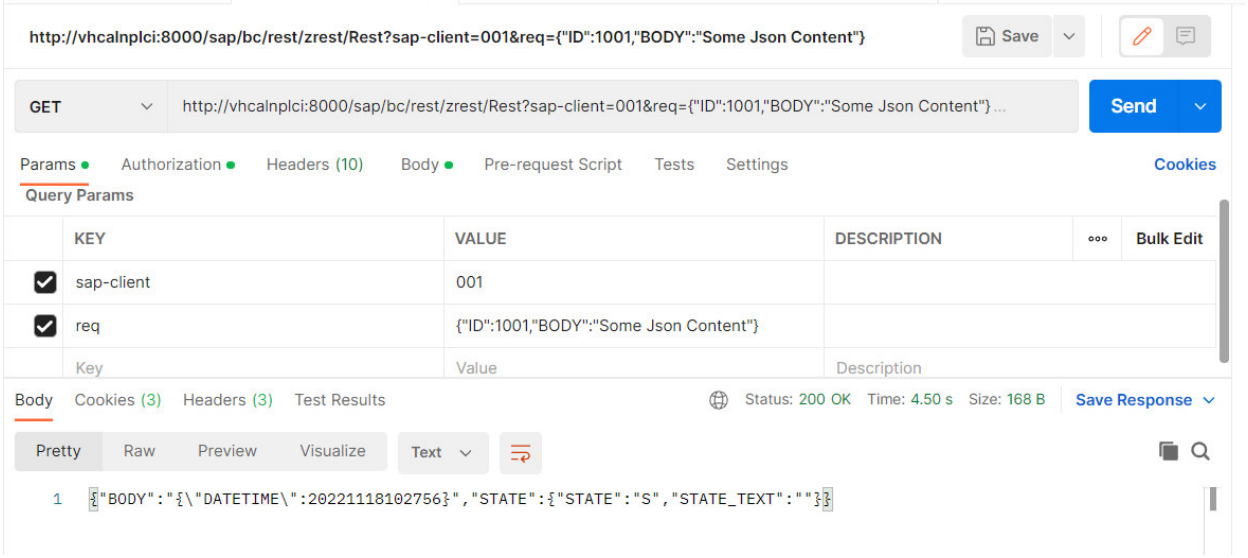
Therefore in my example I have disabled CSRF.

**Postman examples**

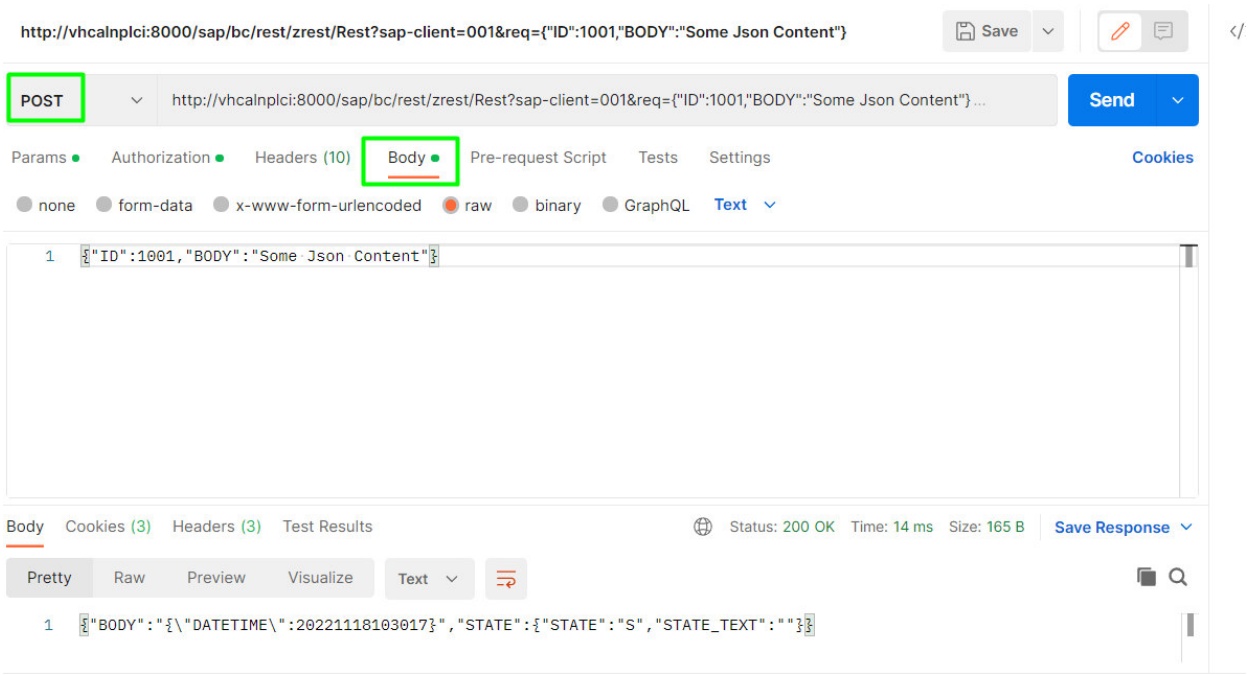
Basic Authentication Parameters




Get Example and Result




Post Example and Result




## To Consume from Abap

We will use a HTTPClient to make call and we will parse json to abap structure with get_datetime example.

Http Client Code

```
CLASS zutil_cl_rest_ws DEFINITION
  PUBLIC
  CREATE PUBLIC .

  "Class wrapper for making Rest Web Service Calls
  PUBLIC SECTION.

    "Constants
    CONSTANTS : c_content_type_json   TYPE string VALUE 'application/
                c_content_type_xml    TYPE string VALUE 'application/
                c_request_method_get  TYPE string VALUE 'GET',
                c_request_method_post TYPE string VALUE 'POST'.

    "Makes WS Call
    METHODS call_ws
      IMPORTING
        VALUE(i_url)            TYPE string
        VALUE(i_content_type)   TYPE string DEFAULT c_content_type_js
        VALUE(i_request_method) TYPE string DEFAULT c_request_method_
        VALUE(i_username)       TYPE string OPTIONAL
        VALUE(i_password)       TYPE string OPTIONAL
        VALUE(i_payload)        TYPE string OPTIONAL
      EXPORTING
        VALUE(e_state)          TYPE zutil_cl_defs=>gty_state
        VALUE(e_response_str)   TYPE string.

  PROTECTED SECTION.

  PRIVATE SECTION.

    DATA http_client TYPE REF TO if_http_client.

    METHODS fill_warning
      RETURNING VALUE(state) TYPE zutil_cl_defs=>gty_state.

ENDCLASS.



CLASS ZUTIL_CL_REST_WS IMPLEMENTATION.
```

```
* <SIGNATURE>-------------------------------------------------------
* | Instance Public Method ZUTIL_CL_REST_WS->CALL_WS
* +-------------------------------------------------------------------
* | [--->] I_URL                          TYPE        STRING
* | [--->] I_CONTENT_TYPE                 TYPE        STRING (default
* | [--->] I_REQUEST_METHOD               TYPE        STRING (default
* | [--->] I_USERNAME                     TYPE        STRING(optional
* | [--->] I_PASSWORD                     TYPE        STRING(optional
* | [--->] I_PAYLOAD                      TYPE        STRING(optional
* | [<---] E_STATE                        TYPE        ZUTIL_CL_DEFS=>
* | [<---] E_RESPONSE_STR                 TYPE        STRING
* +-------------------------------------------------------------------
  METHOD call_ws.
    DATA: exref TYPE REF TO cx_root.
    TRY.

        "Using the this CREATE_BY_URL can simplify certain aspects of
        FREE http_client.
        cl_http_client=>create_by_url(
          EXPORTING
            url    = i_url
          IMPORTING
            client = http_client
          EXCEPTIONS
            argument_not_found = 1
            plugin_not_active = 2
            internal_error    = 3
            OTHERS            = 4
          ).

        IF sy-subrc <> 0.
          e_state-state = fill_warning( ).
          RETURN.
        ENDIF.

        " My logic originally used PUT, but you should be able to cha
        http_client->request->set_method( i_request_method ).
        http_client->request->set_content_type( i_content_type ).


        " Remember to authenticate
        IF i_username IS NOT INITIAL OR i_password IS NOT INITIAL.
          http_client->authenticate(
```

```
    username = i_username
    password = i_password
  ).
ENDIF.


"If exists, prepare payload and assign
IF i_payload IS NOT INITIAL.
  " Convert that payload to xstring.
  DATA lv_payload_x TYPE xstring.
  CALL FUNCTION 'SCMS_STRING_TO_XSTRING'
    EXPORTING
      text   = i_payload
    IMPORTING
      buffer = lv_payload_x
    EXCEPTIONS
      failed = 1
      OTHERS = 2.

  IF sy-subrc <> 0.
    e_state-state = zutil_cl_defs=>c_state_warning.
    e_state-state = 'Encoding error!'.
    RETURN.
  ELSE.
    http_client->request->set_data( lv_payload_x ).    " Bina
  ENDIF.
ENDIF.

" Sending the request
http_client->send(
    EXCEPTIONS
      http_communication_failure = 1
      http_invalid_state         = 2 ).

IF sy-subrc <> 0.
  e_state-state = fill_warning( ).
  RETURN.
ENDIF.


" Receiving the response
http_client->receive(
  EXCEPTIONS
```

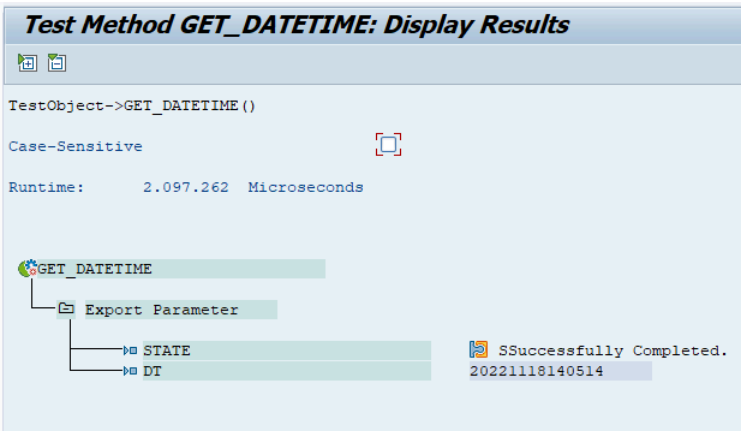
```
            http_communication_failure = 1
            http_invalid_state         = 2
            http_processing_failed     = 3 ).

        IF sy-subrc <> 0.
          e_state-state = fill_warning( ).
          RETURN.
        ENDIF.

        " Check the response. Hopefully you get back a JSON response.
        e_response_str = http_client->response->get_cdata( ).
        IF e_response_str IS INITIAL.
          e_response_str = http_client->response->get_data( ).
        ENDIF.

        e_state-state = zutil_cl_defs=>c_state_success.
        e_state-state_text = 'Successfully Completed.'.

      CATCH cx_root INTO exref.
        e_state-state = zutil_cl_defs=>c_state_error.
        e_state-state_text = exref->get_text( ).
    ENDTRY.
  ENDMETHOD.


* <SIGNATURE>---------------------------------------------------------
* | Instance Private Method ZUTIL_CL_REST_WS->FILL_WARNING
* +--------------------------------------------------------------------
* | [<-()] STATE                          TYPE        ZUTIL_CL_DEFS=>
* +--------------------------------------------------------------------
  METHOD fill_warning.
    state-state = zutil_cl_defs=>c_state_warning.
    http_client->get_last_error(
            IMPORTING
              message        = DATA(msg)" Error Message
          ).
    state-state_text = msg.
  ENDMETHOD.
ENDCLASS.
```

Consumer Class Code. First unwraps the response structure and checks the state. if it is success then unwraps the body json part for result of call id 1001.

```
CLASS zrest_cl_consumer DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    METHODS get_datetime
      EXPORTING
        state TYPE zutil_cl_defs=>gty_state
        dt    TYPE tzntimestp.

  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.



CLASS ZREST_CL_CONSUMER IMPLEMENTATION.


* <SIGNATURE>---------------------------------------------------------
* | Instance Public Method ZREST_CL_CONSUMER->GET_DATETIME
* +--------------------------------------------------------------------
* | [<---] STATE                          TYPE        ZUTIL_CL_DEFS=>
* | [<---] DT                             TYPE        TZNTIMESTP
* +--------------------------------------------------------------------
  METHOD get_datetime.
    "Sample method to consume rest web api
    DATA: exref TYPE REF TO cx_root.
    TRY.
        "
        CONSTANTS: c_uname TYPE string VALUE 'developer',
                   c_pass  TYPE string VALUE 'Down1oad'.

        "Build get call
        DATA: url TYPE string VALUE 'http://vhcalnplci:8000/sap/bc/re

        DATA req_stc TYPE zrest_s_request.
        req_stc-id = '1001'.
        DATA(req_json) = /ui2/cl_json=>serialize( EXPORTING data = re
        url = url && '&req=' && req_json.
```

```
    "Call Web api
    NEW zutil_cl_rest_ws( )->call_ws(
      EXPORTING
        i_url            = url
        i_username       = c_uname
        i_password       = c_pass
      IMPORTING
        e_state          = state
        e_response_str   = DATA(json_response)
    ).

    IF state-state EQ zutil_cl_defs=>c_state_success.

      DATA: resp_stc TYPE zrest_s_response.
      /ui2/cl_json=>deserialize(
                EXPORTING
                  json             = json_response
                CHANGING
                  data             = resp_stc
              ).

      IF resp_stc-state-state EQ zutil_cl_defs=>c_state_success.

        TYPES : BEGIN OF ty_res,
                  datetime TYPE tzntimestp,
                END OF ty_res.
        DATA: resp_1001 TYPE ty_res.

        /ui2/cl_json=>deserialize(
                  EXPORTING
                    json             = resp_stc-body
                  CHANGING
                    data             = resp_1001
                ).


        dt = resp_1001-datetime.
      ENDIF.
    ENDIF.

  CATCH cx_root INTO exref.
    state-state = zutil_cl_defs=>c_state_error.
```

```
        state-state_text = exref->get_text( ).
    ENDTRY.
  ENDMETHOD.
ENDCLASS.
```




*Call Result*

That is all. In that way, you can integrate any environment, system to SAP.

Hope that helps.

And here is a post on BSP based Web Api.

Thanks for reading.

**Related Links**

https://help.sap.com/docs/SAP_NETWEAVER_740/753088fc00704d0a80e7fbd6803c8adb/0f5fb77942744afe94afafa...

https://blogs.sap.com/2013/05/16/usage-of-the-abap-rest-library-sapbasis-740/

https://help.sap.com/docs/SAP_NETWEAVER_740/68bf513362174d54b58cddec28794093/b35c22518bc72214e100000...

https://blogs.sap.com/2022/02/21/creation-of-rest-api-get-post-method-call/

**Tags:**

restapi

Comments

B ziolkowskib Active Contributor 2022 Nov 18 9:47 AM

Hi beyhan.meyrali

Thanks for the post.
You might want to provide links to similar blog posts (i.e. Creation of Rest API (Get & Post Method Call) so people looking for such solution have more examples to choose from.
You might also want to mention if there are any options for monitoring an integration setup based on a custom REST handler and if not what to do to overcome that.

Regards,
Bartosz

 beyhan_meyrali Active Contributor 2022 Nov 18 10:16 AM

Hi Bartosz,

As you have already done in comment, please feel free to add any relevant content to comments.

Thanks for adding links.

A Khan Participant

2023 Feb 08
11:11 AM

Thanks Man, really very nice blog

M 25613 Explorer

2023 Nov 07
8:10 PM

Hello, Thank you for the detailed steps.

beyhan.meyrali  Could you please let me know if the SAP REST api is better or the SAP odata service is better for the use in SAP CPI. We are using the SAP CPI and would like to know the best approach for the interaction of the data.

beyhan_meyrali Active Contributor

2023 Nov 08
7:48 AM

Hi Syed,

Non-SAP world is mostly using Rest based apis. And SAP continues to use oDATA because all tools are developed around oData.

And there might be other reasons to use a middleware tools, such as CPI. One reason might be security. Other might be managing all connections from same place. Other might be separating responsibility and concerns.

Therefore architect must decide what is best solution for each scenario.

N neminath Explorer

2023 Nov 30
2:48 PM

Hi,

I have requirement where I want to expose some services outside my org network. I tried creating OData  but it work only within org network. Outside network (public internet) it is not working.

I have 2 questions :

1. With this method, will I be able to access services outside org network (VPN).

2. If Not, How I can expose API/ODATA which can be used from outside network

Please help.

A alexandreourth Active Participant

2024 Jul 10
5:25 PM
-
edited
2024 Jul 10
5:26 PM

- edited
Very very well documented post.

I used to work with rest service but by directly handling the request inside the 'HANDLE_REQUEST' method of the handler class.

Now thanks to your blog, i'v understood the routing possibilities and by digging more i've found a way to build an API, aiming for one SICF service (so one entry point) and routing the request, according to the extanded path in the **URL**, to the relevant method of data extranction. (IT_PARAMETER of ATTACH method and template with REGEX).

Good work again and thanks for sharing!

Have a good day!